IN THE CLAIMS:

Please amend the claims as follows:

1. (amended) A parallel processor comprising a plurality of processing means which perform mutually parallel processing on the basis of instructions written in a program [programs] and are capable of communicating with each other via a common bus, wherein

A3

one of said processing means suspends processing based on a said program and enters [a] waiting state when executing a wait instruction ("sleep") and releases said waiting state and restarts the processing based on said program based on execution of a wait release instruction ("cont") by another processing means and

the other processing means executes a next instruction without suspending processing after it executes said wait release instruction.

3. (amended) A parallel processor as set forth in claim 1, further comprising:

A4

a first storage means connected to said common bus and storing said programs, and

second storage means provided corresponding to said plurality of processing means, reading from said first storage means programs to be executed by corresponding processing means via said common bus, supplying said processing means with instructions written in the read programs, and having faster access speeds than said first storage means,

[a] said second storage means continuing to store a program supplied to its corresponding processing mean before entering said waiting state when said processing means is in said waiting state.

4. (amended) A parallel processor as set forth in claim 3, wherein [a] said second storage means continues to store a program until its corresponding

A4

5. (amended) A parallel processor as set forth in claim 3, wherein:

said other processing means executes a program execution instruction ("gen") to make said one processing means execute a program stored in said first storage means; and

[a] said second storage means corresponding to said one processing means reads and stores a program which is instructed by said program execution instruction to be executed from said first storage means.

A5

10. (amended) A parallel processor comprising a plurality of processing means which perform mutually parallel processing on the basis of instructions written in a program [program] and are capable of communicating with each other via a common bus, wherein

one of said processing means suspends processing based on [a] said program and enters a waiting state when executing a wait instruction ("sleep") and releases said waiting state and restarts the processing based on said program based on execution of a wait release instruction ("cont_a") by an other [another] processing means and

said other processing means enters a synchronization waiting state when executing said wait release instruction until said one processing means enters said waiting state when said one processing means is not in said waiting state.

A6

12. (amended) A parallel processor as set forth in claim 10, further comprising:

[a] first storage means connected to said common bus for storing said program [programs];

second storage means provided corresponding to said plurality of processing means, reading from said first storage means programs to be executed by corresponding processing means via said common bus, supplying said

processing means with instructions written in the read programs, and having faster access speeds than said first storage means,

[a] said second storage means continuing to store a program supplied to its corresponding processing mean before entering said waiting state when said processing means is in said waiting state.

13. (amended) A parallel processor as set forth in claim 12, wherein [a] said second storage means continues to store a program until its corresponding processing means executes a program end instruction ("end") indicating an end of a program.

14. (amended) A parallel processor comprising a plurality of processing means which perform mutually parallel processing on the basis of instructions written in programs and are capable of communicating with each other via a common bus, comprising:

[a] first storage means connected to said common bus for storing said programs and

second storage means provided corresponding to said plurality of processing means, reading from said first storage means programs to be executed by corresponding processing means via said common bus, supplying said processing means with instructions written in [the] read programs, and having faster access speeds than said first storage means;

one of said processing means suspending processing based on [a] said programs [program] and entering a waiting state when executing a wait instruction ("sleep") and releasing said waiting state and restarting the processing based on said programs [program] based on execution of a wait release instruction ("cont") by other [another] processing means;

[a] said second storage means continuing to store a program supplied to its corresponding processing means [mean] before entering said waiting state when said one processing means is in said waiting state.

15. (amended) A parallel processor as set forth in claim 14, wherein [a] said second storage means continues to store a program until its corresponding processing means executes a program end instruction ("end") indicating an end of a program.

A6

16. (amended) A parallel processor as set forth in claim 14, wherein:

said other processing means executes a program execution instruction ("gen") to make said one processing means execute a program stored in said first storage means; and

[a] said second storage means corresponding to said one processing means reads and stores the program instructed to be executed by said program

A7

18. (amended) A parallel processing method for performing at least first processing and second processing in parallel based on instructions written in programs, wherein:

said first processing suspends processing based on [a] said program and enters a waiting state by executing a wait instruction ("sleep") and releases said waiting state and resumes processing based on said program based on execution of a wait release instruction ("cont") in said second processing and

said second processing executes a next instruction without suspending its processing after executing said wait release instruction.

A8

20. (amended) A parallel processing method for performing at least first processing and second processing in parallel based on instructions written in programs, wherein:

said first processing suspends processing based on [a] said [program] <u>programs</u> and enters a waiting state by executing a wait instruction ("sleep") and releases said waiting state and resumes processing based on said <u>programs</u> [program] based on execution of a wait release instruction ("cont_a") in the second processing, and

said second processing enters a synchronization waiting state by executing said wait release instruction until said first processing enters said waiting state when said first processing is not in said waiting state.

21. (amended) A storage medium for storing in a computer-readable format routines of first processing and second processing to be performed in parallel based on instructions written in programs, wherein:

said first processing is processing which suspends processing based on [a] said program and enters a waiting state by executing a wait instruction ("wait") and releases said waiting state and resumes processing based on said program based on execution of a wait release instruction ("cont") in said second processing, and

said second processing is processing which executes a next instruction without suspending its processing after executing said wait release instruction.

22. (amended) A storage medium for storing in a computer-readable format routines of first processing and second processing to be performed in parallel based on instructions written in programs, wherein:

said first processing is processing which suspends processing based on [a] said program and enters a waiting state by executing a wait instruction ("sleep") and releases said waiting state and resumes processing based on said program based on execution of a wait release instruction ("cont_a") in the second processing, and

said second processing is processing which enters a synchronization waiting state by executing said wait release instruction until said first processing enters said waiting state when said first processing is not in said waiting state.

R E M A R K S